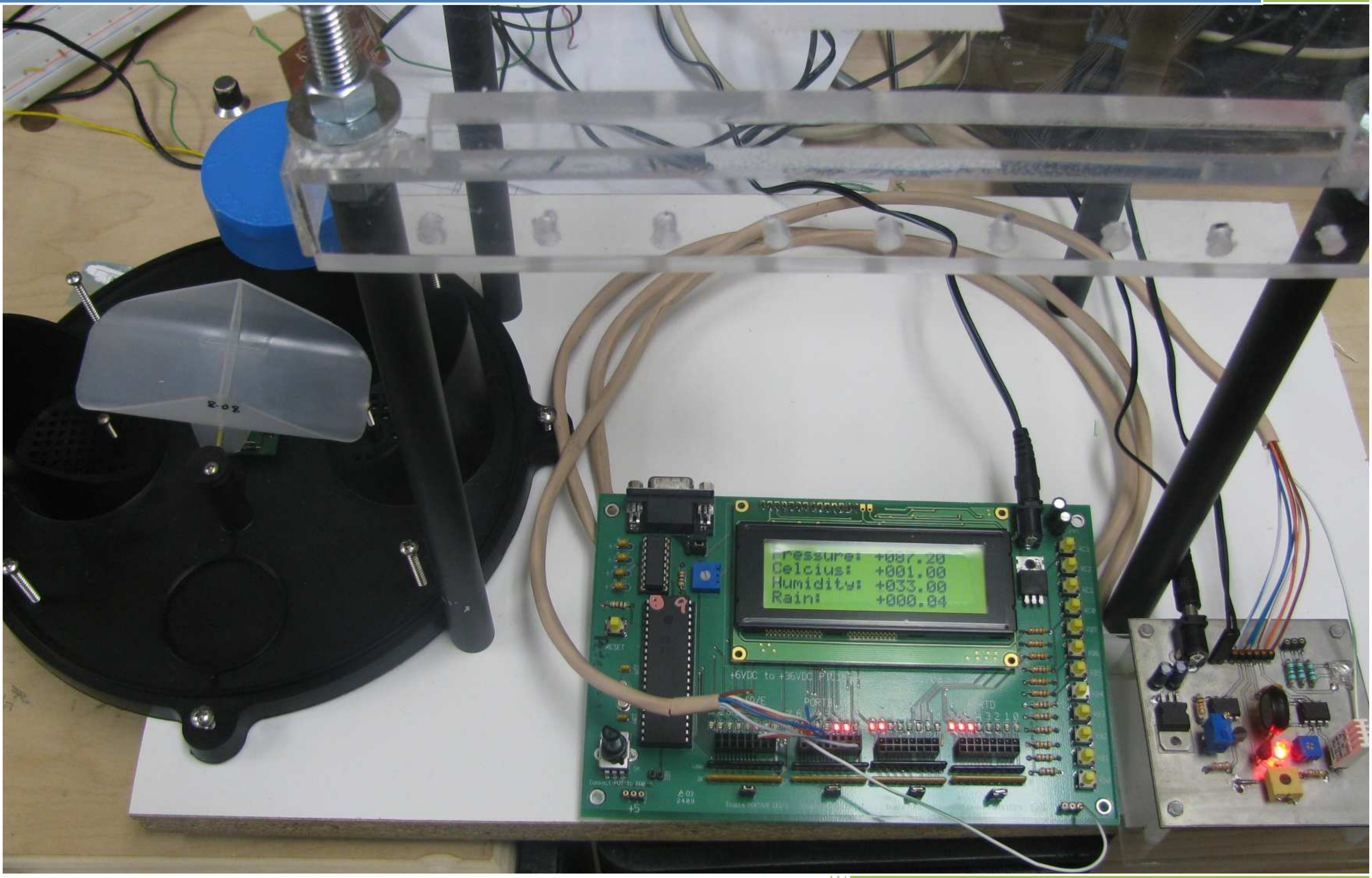
Weather Station for RFID Study



Dan Peterson

Ashish Roy

Divyanshu Narendra

Advisor: Dr. Glower

SD0920

Table of Contents:

Background:

The purpose of this weather station is to measure and record the weather elements while testing the sensitivity of RFID tags. RFID tags are planning on being used for cattle management. However, a good tag design has to be created for this process to take place. The tags that are fitted to the cattle have to withstand the elements of the weather. To create the ideal tag, weather testing on possible tag designs has to be done while recording the weather elements and analyzing how the weather affects the sensitivity and durability of the tag designs.

Previous tag testing has been done while fitted on cattle, but a more in depth study has to be performed to perfect this method. A study of weather effects on RFID tags is needed to prove the viability of RFID technology for cattle. This is where our project design comes in to design a portable weather station to be used in this study.

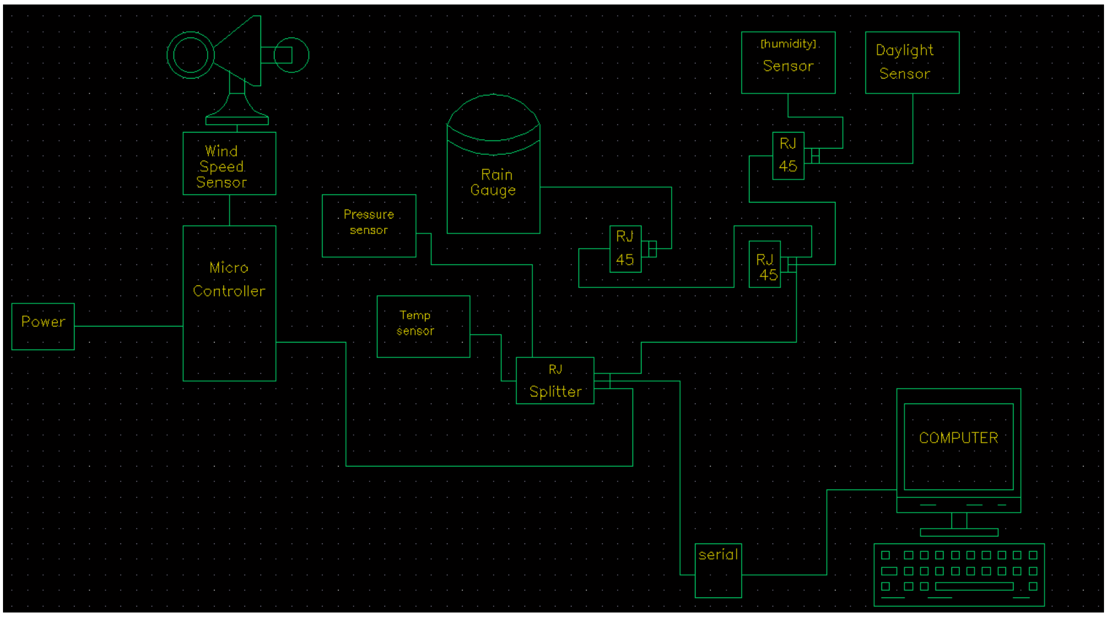
Requirements:

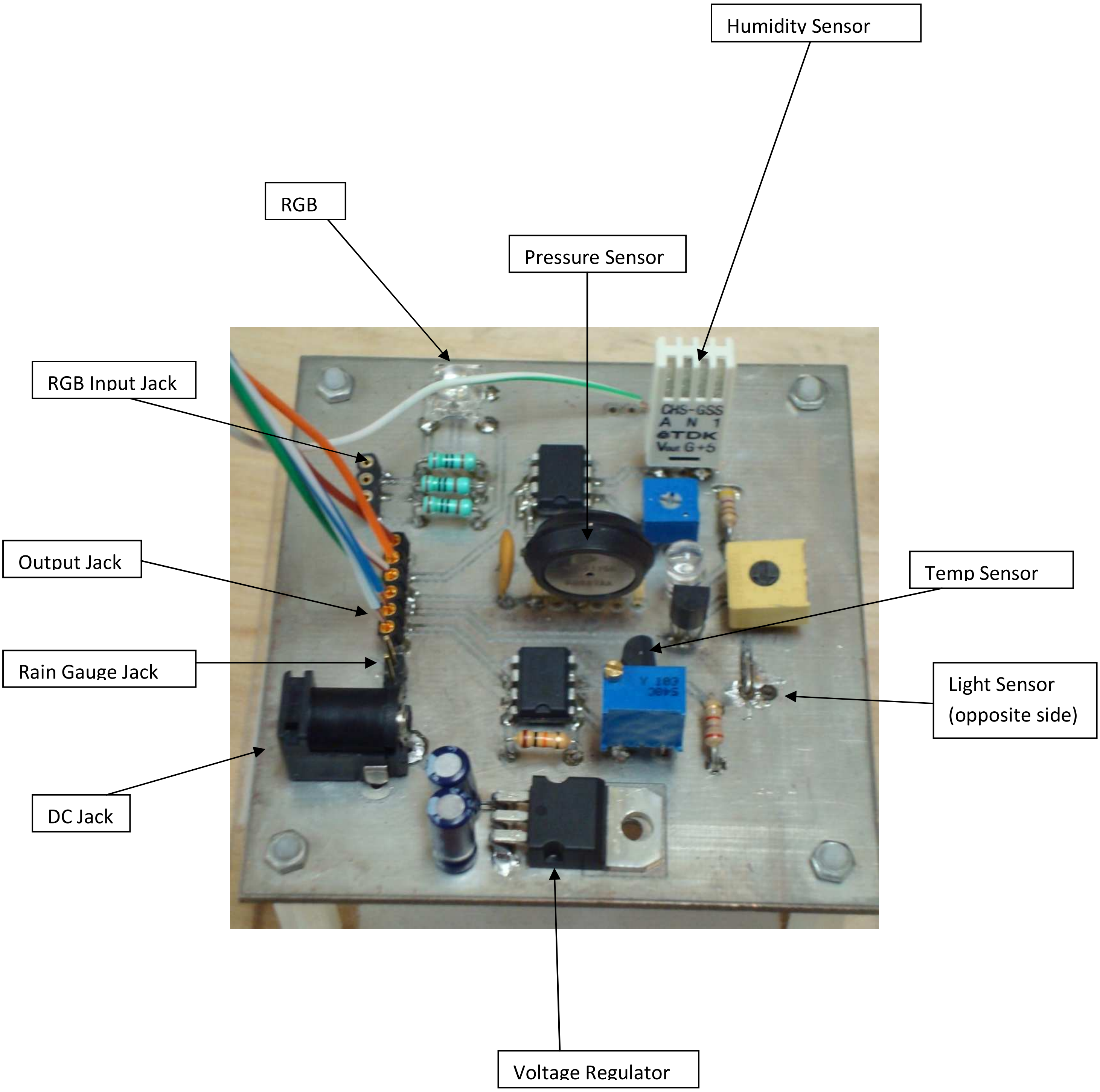
We were required to design this weather station and record the following weather elements.

- Temperature
- Atmospheric Pressure
- Rainfall
- Daylight
- Humidity
- Wind speed

While these elements are being observed, we needed to be able to capture and record all the incoming data. We have designed a system to send all our data back to a PC and the data can be recorded in a text file to be used for further analysis. Our design has allowed data to be recorded for an extended period of time without needing to be under constant observation. The data can then be used with the RFID sensitivity test data to further explore the possibilities of RFID technology for cattle management.

Block Diagram




Humidity Sensor
RGB
Pressure Sensor
RGB Input Jack
Output Jack
Temp Sensor
Rain Gauge Jack
Light Sensor
(opposite side)
DC Jack
Voltage Regulator

Atmospheric Pressure Sensor: Motorola MPX4115A

The MPX4115A is an integrated silicon pressure sensor for manifold absolute pressure, altimeter or barometer applications. This sensor integrates on-chip, bipolar op amp circuitry and thin film resistor networks to provide a high output signal and temperature compensation.

The MPX4115A piezoresistive transducer is a state–of–the–art, monolithic, signal conditioned, silicon pressure sensor. This sensor combines advanced micromachining techniques, thin film metallization, and bipolar semiconductor processing to provide an accurate, high level analog output signal that is proportional to applied pressure.

Features:

- 1.5% maximum error over 0°C to 85°C
- Ideally suited for microprocessor or microcontroller based systems
- Temperature compensated from -40°C to +125°C
- 15 to 115 kPa (2.2 to 16.7 psi)
- Supply Voltage 4.85-5.35 VDC
- Durable epoxy unibody element

Schematics:

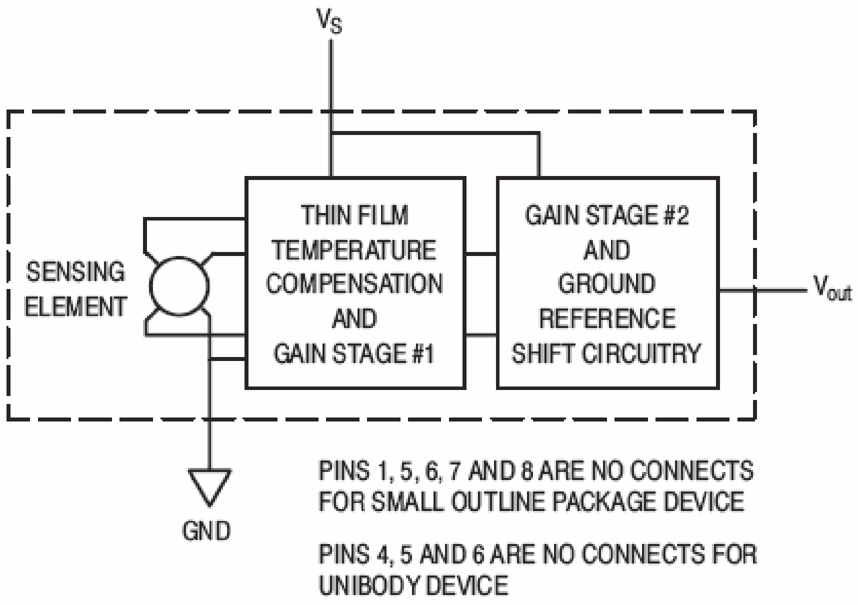


Figure 1. Fully Integrated Pressure Sensor Schematic

Figure 1 shows a block diagram of the internal circuitry integrated on a pressure sensor chip.

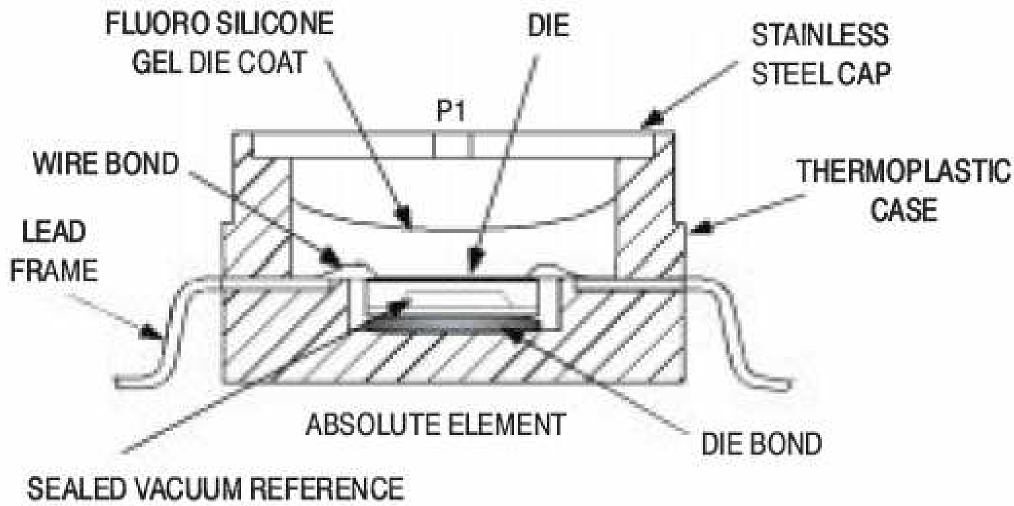


Figure 2. Cross Sectional Diagram SOP (not to scale)

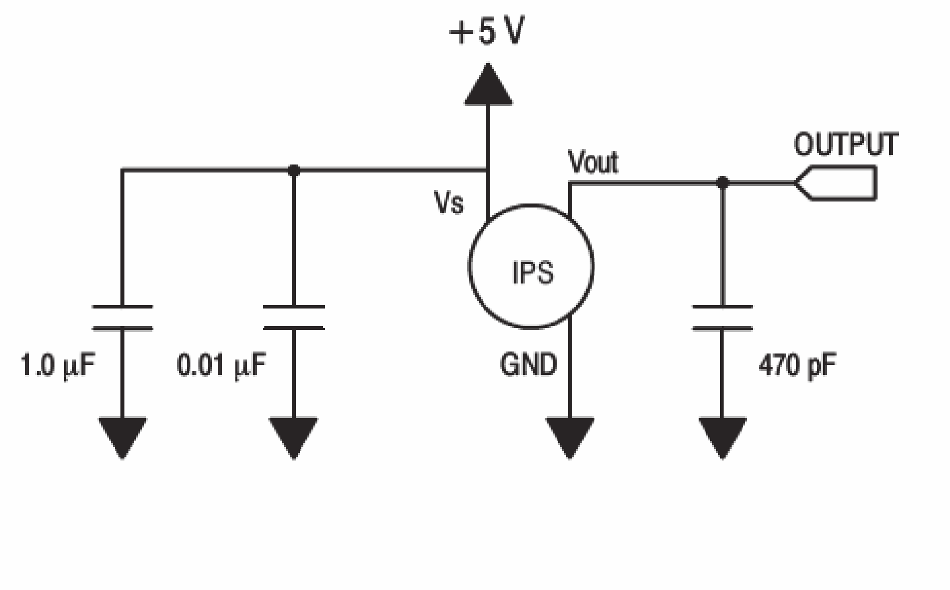
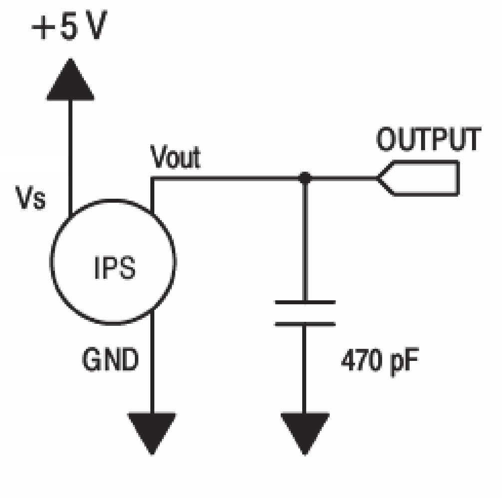


Figure 3. Recommended power supply decoupling and output filtering.

Figure 3 is recommended for a circuit without designated ground and power planes. Our PCB contains decoupling capacitors between the power plane and ground plane already so we did not need to include the two capacitors on the left side of the circuit. Our circuit diagram appears below.



PIN NUMBER			
1	V_{out}	4	N/C
2	Gnd	5	N/C
3	V_S	6	N/C

NOTE: Pins 4, 5, and 6 are internal device connections. Do not connect to external circuitry or ground. Pin 1 is noted by the notch in the lead.

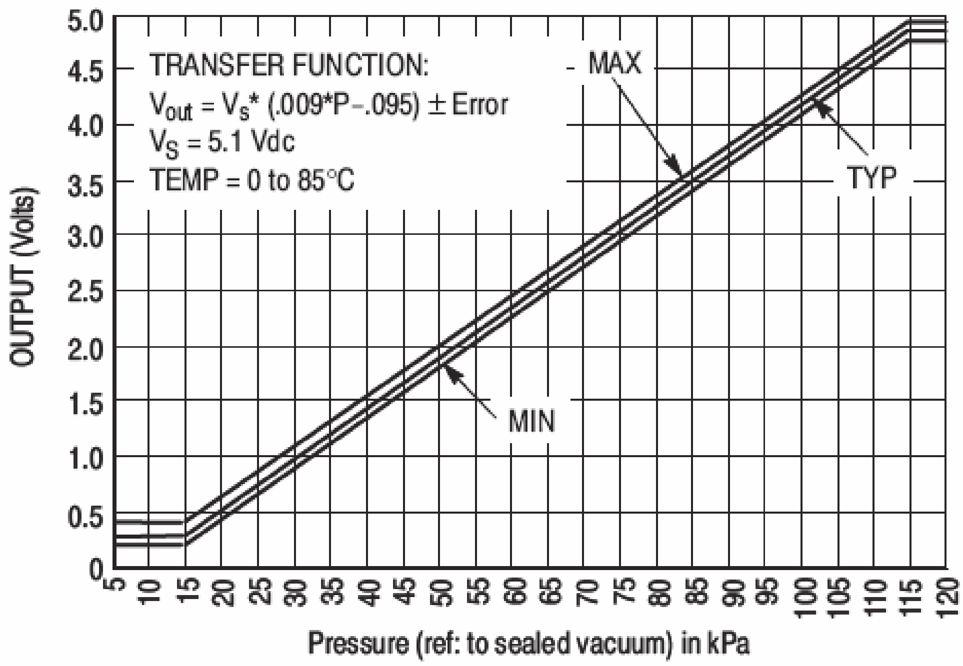
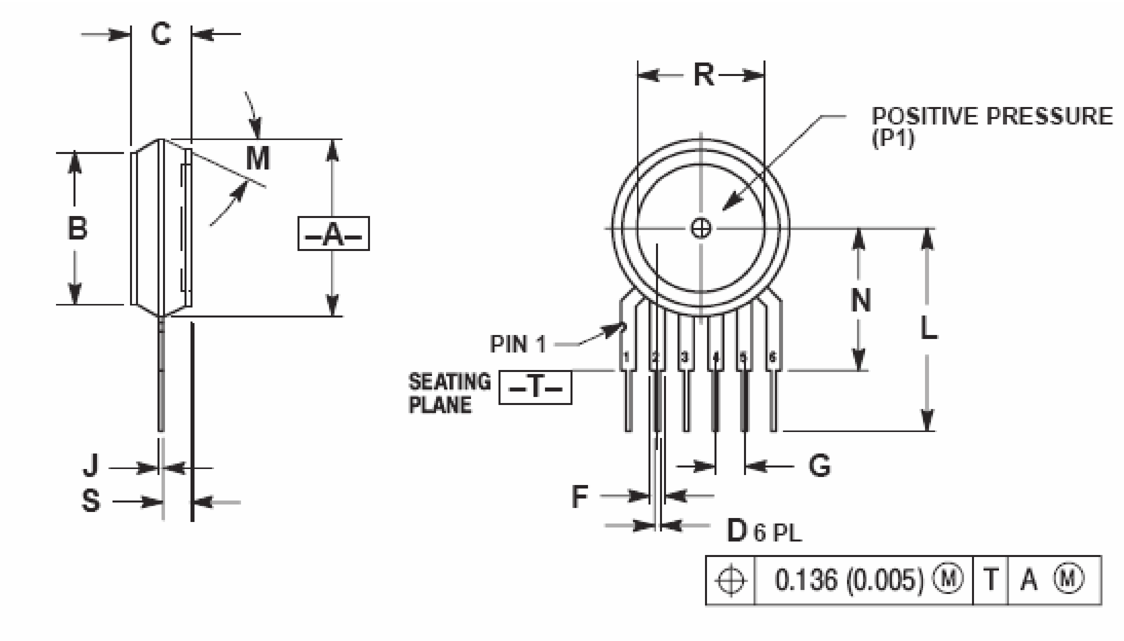


Figure 4. Output versus Absolute Pressure

Figure 4 shows the sensor output signal relative to the pressure input. Typical minimum and maximum output curves are shown for operation over 0°C to 85°C temperature range. The output will saturate outside of the rated pressure range. A fluorosilicone gel insolates the die surface and wire bonds from the environment, while allowing the pressure signal to be transmitted to the silicon diaphragm.



DIM	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	0.595	0.630	15.11	16.00
B	0.514	0.534	13.06	13.56
C	0.200	0.220	5.08	5.59
D	0.027	0.033	0.68	0.84
F	0.048	0.064	1.22	1.63
G	0.100 BSC		2.54 BSC	
J	0.014	0.016	0.36	0.40
L	0.695	0.725	17.65	18.42
M	30 ° NOM		30 ° NOM	
N	0.475	0.495	12.07	12.57
R	0.430	0.450	10.92	11.43
S	0.090	0.105	2.29	2.66

Notes:

1. Dimensioning and tolerancing per ansi Y14.5M, 1982.
2. Controlling dimension: inch
3. Dimension –A– is inclusive of the mold stop ring. Mold stop ring not to exceed 16.00 (0.630).

Software:

The pressure sensor software uses analog to digital conversion, serial port communication, and the LCD display function. Refer to the entire code listed in the Appendix for the A2D_Init, A2D_Read, SCI_Init, SCI_Out, and LCD_Out subroutines.

In the main routine, we designated PRESSURE as an unsigned integer (unsigned int PRESSURE) and set it at a value of 0 (PRESSURE = 0). The pressure sensor is running into input RA2 on the board so we set A2D2 = A2D_Read(2). We now used the transfer function given for the sensor.

$$V_{out} = VS \times (0.009 \times P - 0.095)$$

With VS being our source of +5V, we solved this equation for P and ended up with this equation:

$$P = 22.22 \times V_{out} + 10.56$$

By substituting Vout = A2D x 0.488 to cover the A2D 5 volt spread, we end up with the following equation:

$$P = 22.22 * A2D * 0.488 + 10.56$$

Similar to the pressure value, we designated VOLT as the output voltage of the sensor and initialized it to 0. In our while loop of our main routine, we include the functions we just calculated:

VOLT = A2D2*.488
PRESSURE = 22.22*VOLT + 10.56

We displayed the pressure in kPa by using:

```
LCD_Move(0,10)
LCD_Out(PRESSURE)
```

Finally, we send the calculated value of PRESSURE back to the PC for data collection by using the line of code:

```
SCI_Out(PRESSURE, 2)
```

The comma 2 after PRESSURE puts the decimal for the value to be read correctly.

RainWise Wired Digital Rain Gauge:

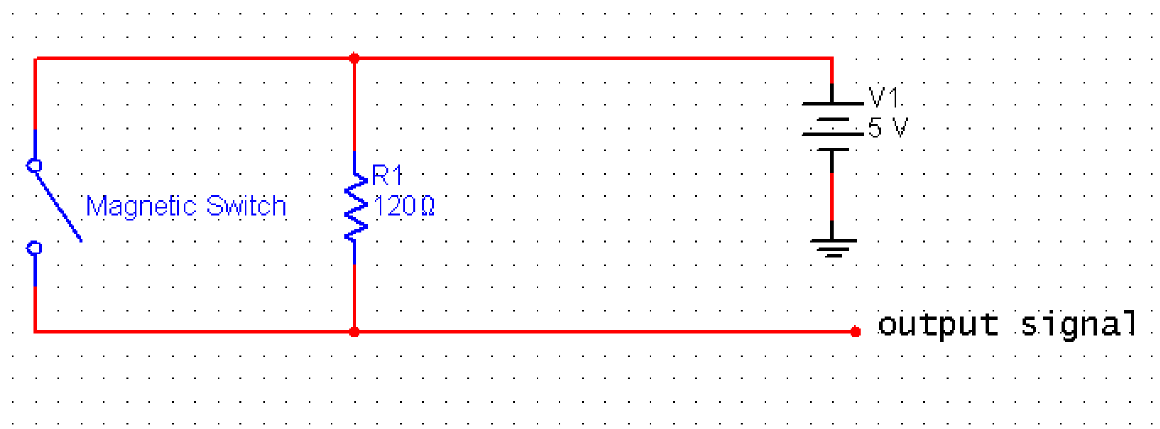
The RainWise 8” diameter collector meets National Weather Service specifications for statistical accuracy. Every time the bucket tips, a count is transmitted to the indoor display and the gauge empties. You never have to empty the gauge. Each tip of the bucket shows one hundredth of an inch on the indoor LCD display. The display will indicate up to 9.99” of Rain. A button on the counter lets you reset to zero after each rainfall

For our purposes, we will not use the included LCD digital counter. Instead, we have made it possible to connect the 60 ft cable to our PCB. The connection is

the two pin jumper located between the DC power jack and the output jacks. We have tied the rain gauge into our PCB design so we can have the output sent to our PIC and then sent to the PC for data collection. By this method, we will be able to keep track of much more rainfall before resetting.

The rain gauge works on a simple circuit. One lead is connected to a voltage supply and the other is connected to the counter. In our design, we connected the power lead to our common 5 volt power supply and the other to an output pin that runs to input RB0 on our embedded board. The rest of the circuit contains a 125Ω resistor and magnetic switch in parallel with the leads. When the bucket tips, the magnetic switch is tripped for a moment, and the circuit shorts. This causes the supply voltage to short to the counter. A diagram of this circuit is shown below.

Schematic:



Software:

This part of our program also uses serial port communication and the LCD display function. Refer to the entire code listed in the Appendix for the SCI_Init, SCI_Out, and LCD_Out subroutines.

At the beginning of our code, we designated RAIN as an unsigned integer with the line:

```
unsigned int RAIN
```

We use a certain subroutine to keep track of our RAIN value called an interrupt. The following is the subroutine for this interrupt we used:

```
void interrupt IntServe(void) {
        if (INT0IF) {
          if (RB0) RAIN += 1;
          INT0IF = 0;
        }
}
```

We are using the INT0 interrupt here. Basically what is happening is that every time a pulse is sent to our input RB0, our integer RAIN is increased by one. We are using the flag INT0IF for this interrupt. We also need to enable this interrupt in our program by including the following:

```
TRISB0 = 1
INT0IE = 1
GIE = 1
PEIE = 1
```

The last bit we need to include to enable is:

```
INTEDG0 = 1
```

This line enables our interrupt to count the rising edge of the pulse.

What we include in our while loop now is similar to the pressure section as we also display it to the LCD screen using:

```
LCD_Move(3,10)
LCD_Out(RAIN)
```

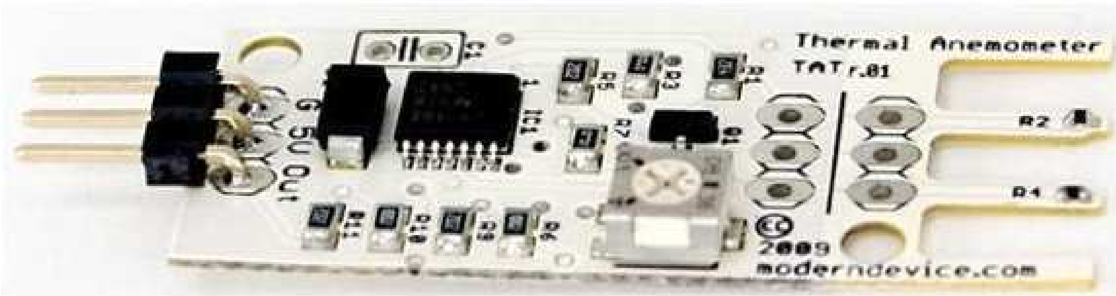
The 3 means we display RAIN third from the top on our LCD display. Finally, we send our counted integer RAIN back to the PC using the command:

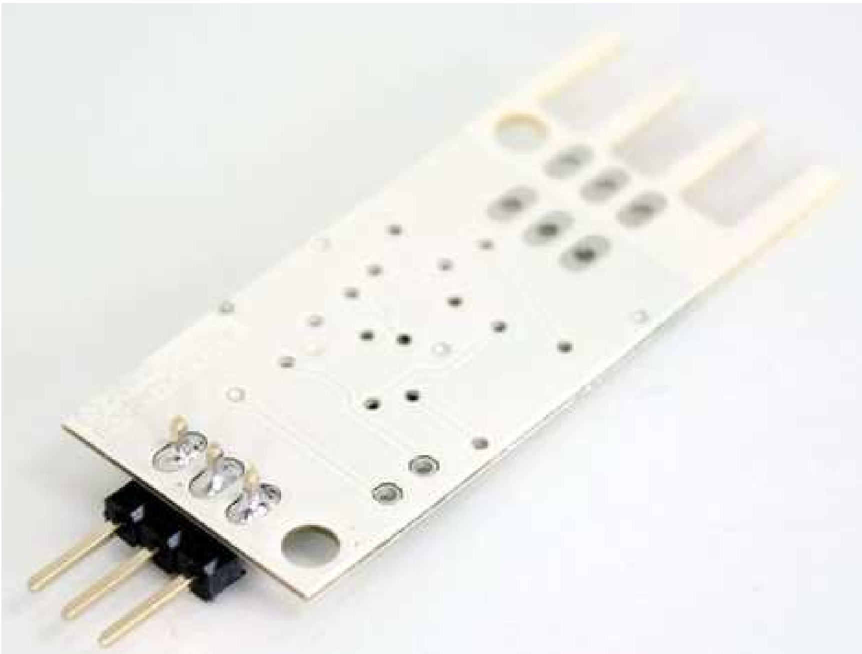
```
SCI_Out(RAIN, 2)
```

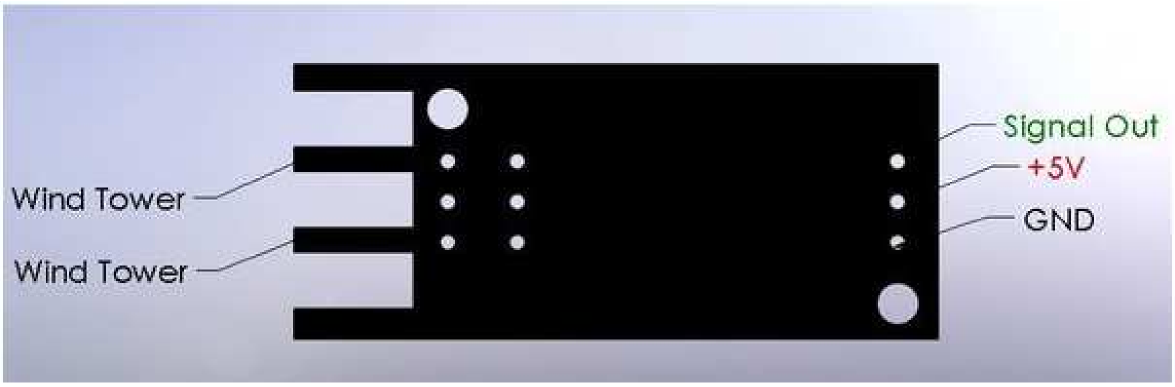
The comma 2 puts the decimal in the correct spot to show that the rain is counting in hundredths of an inch.

Wind Speed Sensor: Thermal Anemometer Thermistor

After failed attempts at creating our own wind speed sensor circuit, we decided to order this one from Modern Device. This product is based on the same principle that our design was.







This sensor circuit is a thermal anemometer based on a traditional technique for measuring wind speed. The technique is called the "hot-wire" technique and involves heating an element to a constant temperature and then measuring the electrical power that is required to maintain the heated element at that temperature as the wind changes. The measured electrical input is then directly proportional to the wind speed.

This sensor is extremely sensitive, as a small puff of air can be sensed at a distance of 1.5-2 feet. At start up, the sensor needs to be allowed to warm up for about 10 seconds to thermally stabilize for best accuracy.

The wind sensor includes a small trimpot that is used to calibrate the sensor for zero wind. The sensor needs to be calibrated before accurate wind speed measurements can be made. To calibrate, simply place a glass over the sensor to block any air movement and adjust the pot for the desired zero level. With a supply voltage of 5 Volts, adjust the lower calibration point to 0.2-0.5 Volts. A lower calibration point will result in a little more sensing capability at the high end.

The output of the sensor is logarithmic. This means the sensor can capture very slight air movements at the low end but also not saturate at full output until the air-flow reaches about 30 mph.

Specifications:

- Dimensions: .68" x 1.59" x .25"
- Supply voltage: 4-12 Volts (designed and calibrated at the factory for 5V)
- Supply current: 20-40 mA (depending on wind speed)
- Output signal: Analog, 0 to VCC

This sensor did not arrive in time for us to be able to test it and write a program for it, so there is work to be done to add this to our PCB. However, we have included a header in our PCB for this sensor to be added to. The header is made so that the signal out runs to the output jack, the middle pin runs to the 5V

power plane, and the other pin runs to the ground plane. It simply needs to be soldered into place.

The sensor just has to be placed in the right direction with the thermistors on the wind towers pointed out. If placed like this, the sensor will be lined up with the PCB footprint correctly. It would also be easy to line it up correctly by following the traces on the PCB. Simply line up the labeled signal out pin on the sensor with the trace that leads to the output jack.

Once the sensor is added to the PCB, a subroutine has to be added to our program. We were not fortunate enough to be able to test this sensor and create a subroutine, so there is future work to be done. This can be done by using the methods we implemented in our software to write an integer value to the LCD display and to send data back to the PC through the serial port for our other sensor circuits. However, we unfortunately cannot give complete instructions for the transfer function and how to read the output signal. A function will have to be created based on testing and the information we have previously given in the specifications and details of this sensor.

Temperature sensor:

LM335:

- DIRECTLY CALIBRATED IN 1°C
- INITIAL ACCURACY
- OPERATES FROM 450µA TO 5mA
- LESS THAN 1ΩDYNAMIC IMPEDANCE
- WIDE OPERATING TEMPERATURE RANGE
- LOW COST

DESCRIPTION

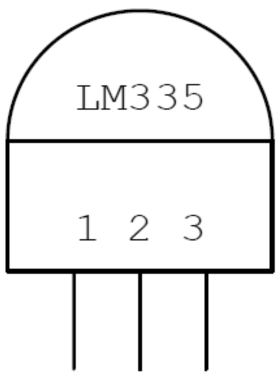
The LM335 is precision temperature sensor which can be easily calibrated. Operates as a 2-terminal Zener and the breakdown voltage is directly proportional to the absolute temperature at 10mV/°K. The circuit has a dynamic impedance of less than 1Ω and operates within arrange of current from 450µA to 5mA without alteration of its characteristics. Calibrated at +25 C, the LM335 have a typical error of less than 1oC over a 100°C temperature range. Unlike other sensors LM335 have a linear output.

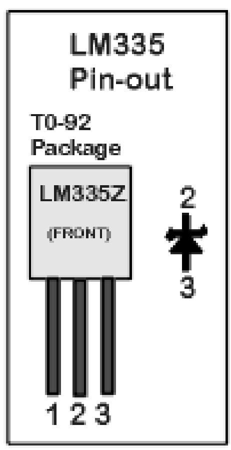
Pin Out Diagram

1 = ADJ

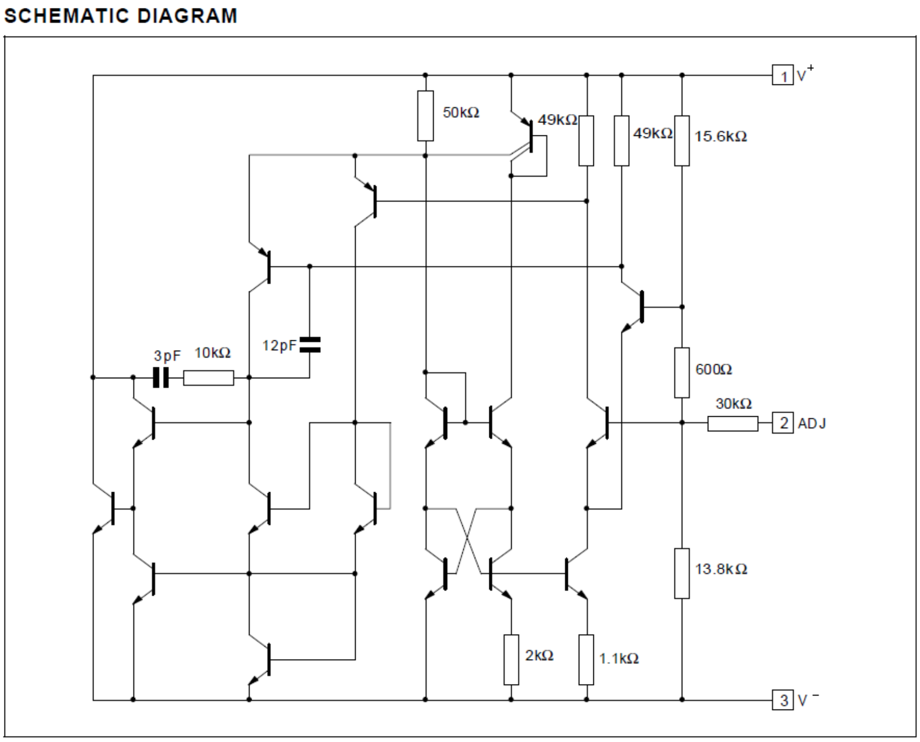
2 = V+

3 = V-





The interior schematic of the LM 335 chip:



ADDITIONAL INFO.

There is an easy method of calibrating the device for higher accuracies (see typical applications). The single point calibration works because the output of the LM135, LM235, LM335 is proportional to the absolute temperature with the extrapolated output sensor going to 0V at 0°K (–273.15oC).Errors in output voltage versus temperature are only slope. Thus a calibration of the slope at one temperature corrects errors at all temperatures.

The output of the circuit (calibrated or not) can be given by the equation: VOT = VOTO xT/To where T is the unknown temperature and TO is the reference temperature (in oK).Nominally the output is calibrated at 10mV/°K. Precautions should be taken to ensure good sensing accuracy. As in the case of all temperatures sensors, self heating can decrease accuracy. TheLM135, LM235, LM335 should operate with a low current, but sufficient to drive the sensor and its calibration circuit to their maximum operating temperature. If the sensor is used in surroundings where the thermal resistance is constant, the errors due to self heating

Can be externally calibrated. This is possible if the circuit is biased with a temperatures table current. Heating will then be proportional to Zener voltage and therefore temperature. In this way the error due to self heating is proportional to the absolute temperature as scale factor errors.

Temperature Accuracy LM335, LM335A (Note 1)

Parameter	Conditions	LM335A			LM335			Units
		Min	Typ	Max	Min	Typ	Max	
Operating Output Voltage	$T_C = 25°C$, $I_R = 1$ mA	2.95	2.98	3.01	2.92	2.98	3.04	V
Uncalibrated Temperature Error	$T_C = 25°C$, $I_R = 1$ mA		1	3		2	6	°C
Uncalibrated Temperature Error	$T_{MIN} \le T_C \le T_{MAX}$, $I_R = 1$ mA		2	5		4	9	°C
Temperature Error with 25°C Calibration	$T_{MIN} \le T_C \le T_{MAX}$, $I_R = 1$ mA		0.5	1		1	2	°C
Calibrated Error at Extended Temperatures	$T_C = T_{MAX}$ (Intermittent)		2			2		°C
Non-Linearity	$I_R = 1$ mA		0.3	1.5		0.3	1.5	°C

Electrical Characteristics (Note 1)

Parameter	Conditions	LM135/LM235 LM135A/LM235A			LM335 LM335A			Units
		Min	Typ	Max	Min	Typ	Max	
Operating Output Voltage Change with Current	400 $\mu A \le I_R \le 5$ mA At Constant Temperature		2.5	10		3	14	mV
Dynamic Impedance	$I_R = 1$ mA		0.5			0.6		Ω
Output Voltage Temperature Coefficient			+10			+10		mV/°C
Time Constant	Still Air 100 ft/Min Air Stirred Oil		80 10 1			80 10 1		sec sec sec
Time Stability	$T_C = 125°C$		0.2			0.2		°C/khr

OBJECTIVE:

Design the Temperature Sensor for a Range of 0°C to 50°C

VOUT range is 0°C (2.73V) to 100°C (3.23V).
Let IR = 450µA (above the Min.) to keep Heat Dissipation in Sensor Low.
R1 = (5V – 3.73V)/450µA = 2.82KΩ so select 2.7K

CHECK:
IR(min) = (5-2.73)/2.7K = 470µA OK and
IR(max) = (5-3.73)/2.7K = 840µA OK
Let RPOT = 10K as recommended.

INTERFACE TO ADC

Need to amplify and shift VTEMP (2.73V – 3.3V) to (0 to 5V) for ADC Full Dynamic Range. Requires use of a DC Offset and Amplifier to condition signal.

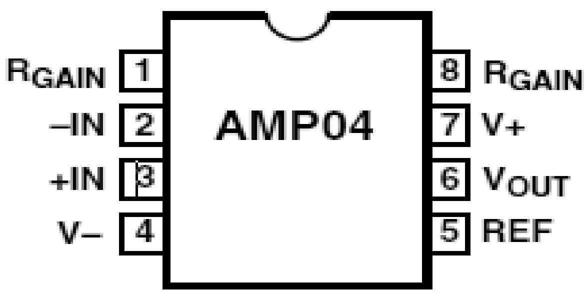
VOLTAGE GAIN

- ΔVTEMP = ΔVIN = 3.23V – 2.73V = 0.50V
- ΔVADC = ΔVOUT = 0V – 5.00V = 5.00V
- AV = ΔVOUT/ΔVIN = 10

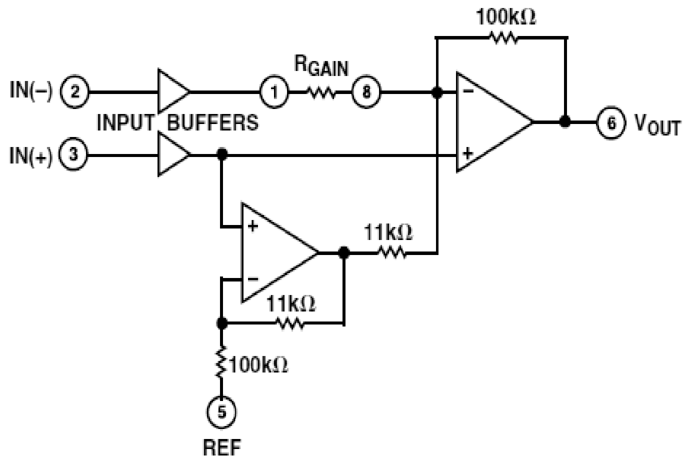
DC OFFSET:

From the data sheets, this isn't obvious what that offset should be. Used a Potentiometer so that anything between 0V and 5V can be obtained

AMP04 INSTRUMENTATION AMPLIFIER:

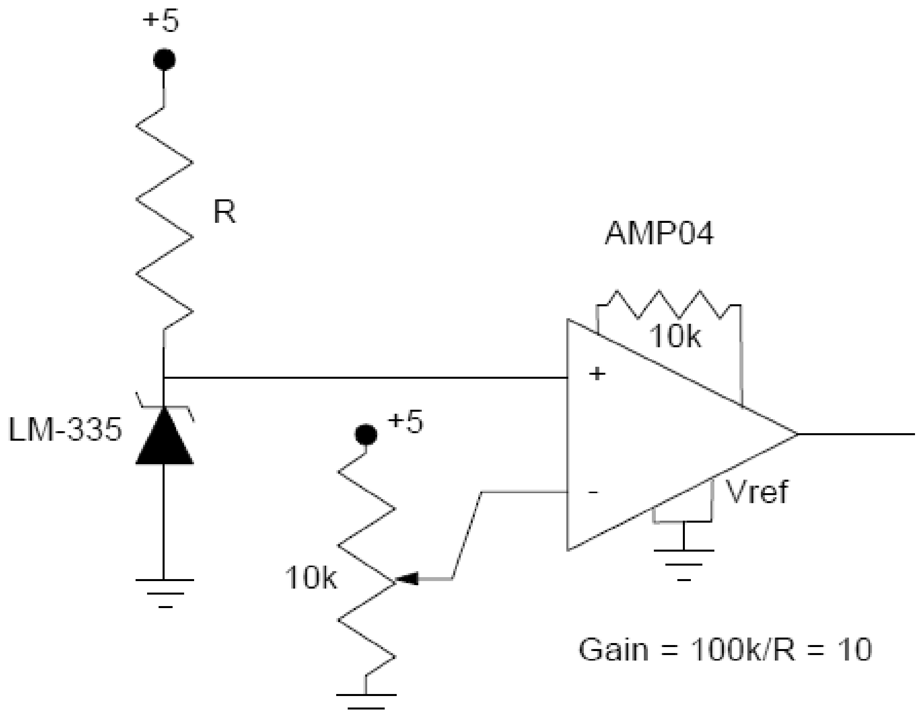


FUNCTIONAL BLOCK DIAGRAM



- R1 = 100k
- R2 = 11k
- Rg = external resistor
- The input buffers have an input impedance on the order of 4*GW, so this chip will not load the sensor very much.*
- The gain is easily adjusted by varying Rg - an external resistor

THE FINAL CIRCUIT:



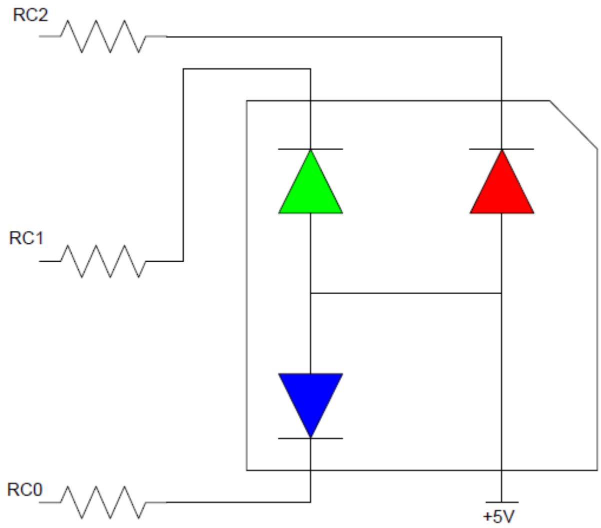
SOFTWARE :

```
void main (void)
{
  unsigned char i, j;
  unsigned int A2D,A2D0
  unsigned int CELCIUS;


  CELCIUS = 0;


While (1) {

    A2D0 = A2D_Read(0);
```

```
    CELCIUS=A2D0*0.048;

    //Display Celsius
    LCD_Move(1,10);
    CELCIUS = (CELCIUS)*100;
    Display(CELCIUS);

    SCI_Out (CELCIUS, 2);
    SCI_CRLF();


    Wait_ms(1000);
}
```

USE OF COLOR WHEEL LED:

SCHEMATIC AND INTERFACE TO PIC:



PURPOSE:

- Create pretty colors with a PIC
- Showcase using pulse width modulation (PWM) to vary the output voltage
- Use the LCD display to convey information.

OBJECTIVE OF USING COLOR WHEEL

- Cold temperature is indicated by BLUE LED
- Average temperature is indicated by GREEN LED
- Hot temperature is indicated by RED LED

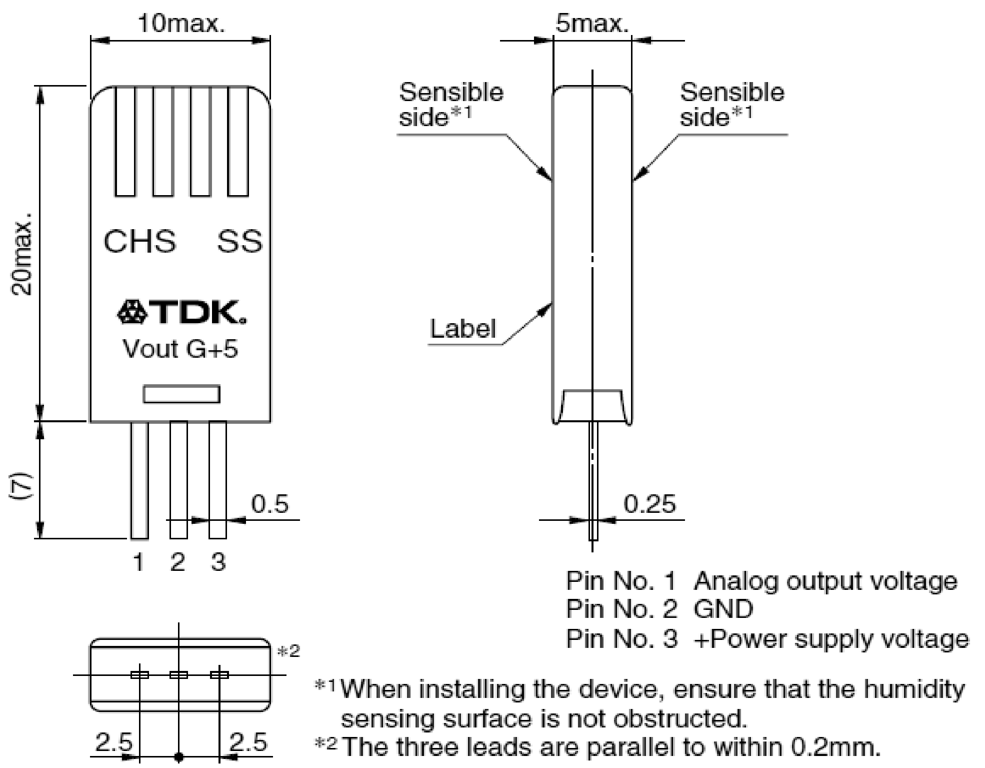
PROGRAM FOR COLOR WHEEL:

```
      if (CELCIUS < 15)
    {
                      BLUE = 100;
                      RED = 0;
                      GREEN = 0;

       if ((CELCIUS >= 15) & (CELCIUS < 25) )
    {
          GREEN = 10*(CELCIUS - 15);
          BLUE = 100 - GREEN;
          RED = 0;
     }
       if ((CELCIUS >= 25) & (CELCIUS < 40) )
     {
          RED = 10*(CELCIUS - 25);
          GREEN = 100 - RED; BLUE = 0;
     }
       if (CELCIUS >= 40)
    {
         RED = 100;
         GREEN = 0;
         BLUE = 0;
     }
     for (j=0; j<100; j++)
    {
     for (i=0; i<100; i++)
     {
         if (RED > i) RC0 = 0; else RB1 = 1;
         if (GREEN > i) RC1 = 0; else RB2 = 1;
         if (BLUE > i) RC2 = 0; else RB3 = 1;
     }

    }

    }
```

HUMIDITY SENSOR: CHS GSS

- Characteristics are stable over a wide temperature range.
- Humidity sensing characteristics exhibit virtually no hysteresis.
- Highly cost-effective and compact, requiring extremely little Mounting space.
- Low current consumption.
- Outputs DC.1V at 100(%)RH; relative humidity can be read directly with a voltmeter.
- All-in-one construction integrates sensor with support circuitry. The entire module operates off a 5V power supply.
- Generated ripple at low humidity levels will not exceed 2.5mV.

TDK's CHS series humidity sensors are compact and extremely simple to apply. Because they contain the necessary circuitry, there is no need to provide additional control circuitry or perform time-consuming calibration. With simple connection to a power supply, they will output DC at 100% relative humidity. This makes it possible to read RH directly with a voltmeter.



ELECTRICAL CHARACTERISTICS

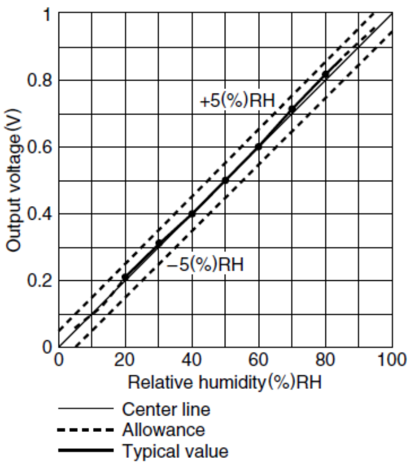
Item	Specifications			Conditions
	Minimum	Standard	Maximum	
Operating voltage Edc (V)	4.75	5	5.25	
Operating current(mA)			0.6	Edc=5V, 25°C
Output voltage(mV)/(%)RH		10		Edc=5V, 25°C
Output impedance(kΩ)		(200)*		at DC
Accuracy(%)RH	–5		+5	Edc=5V, 25°C, 20 to 85(%)RH(For details, please refer to typical characteristics)
Hysteresis(%)RH		≈0		Stable time: 20min
Temperature dependency(%)RH	–5		+5	Edc=5V, 25°C standard
Response time(min)		1		Response time to reach 90% of actual humidity as for from 30 to 85(%)RH
Recommended operating temperature(°C)	+15		+35	Edc=5V, without dewing(For details, please refer to typical characteristics)

*( ): Reference value

CHS-SS TYPE
TYPICAL CHARACTERISTICS

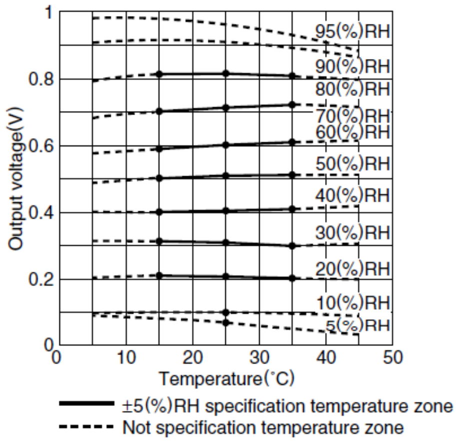
SENSOR LINEARITY CHARACTERISTICS (Ta=25°C Edc=5V)

CHS-MSS TYPE



TEMPERATURE DEPENDENCY CHARACTERISTICS

CHS-MSS TYPE



ADDITIONAL INFO

From datasheet it is clear that we are getting output from 0 to 1 v dc we need to amplify it to get 0 to 5 volt spread

VOLTAGE GAIN

- ΔVHUMIDITY = ΔVIN = 0V – 1V = 1V
- ΔVADC = ΔVOUT = 0V – 5.00V = 5.00V
- AV = ΔVOUT/ΔVIN = 5V

Amplifier used is AMP04 details mentioned above.

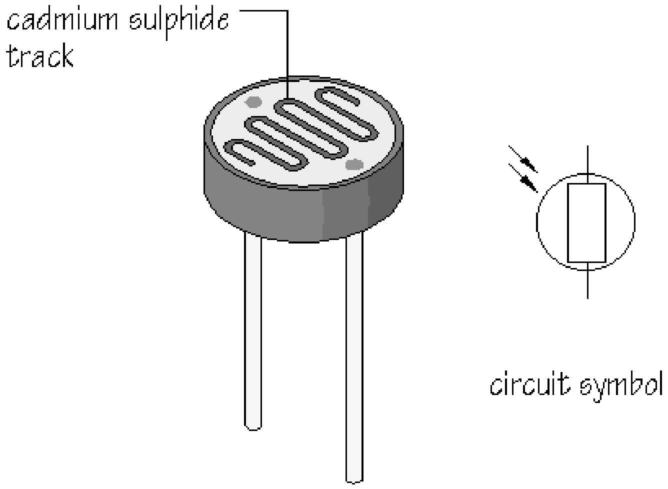
Circuit will be similar to that of temperature circuit and we don't hav to use any resistor between the power ground and pin 1 of the sensor and the pot should not be connected to the power supply

SOFTWARE FOR HUMIDITY:

```
const unsigned char MSG2[21] = "Humidity:         ";

// Subroutine Declarations
#include      <pic18.h>
#include      "lcd_portd.h"
#include      "function.h"

// Subroutines
#include      "lcd_portd.c"
#include      "function.c"


void main(void)
```

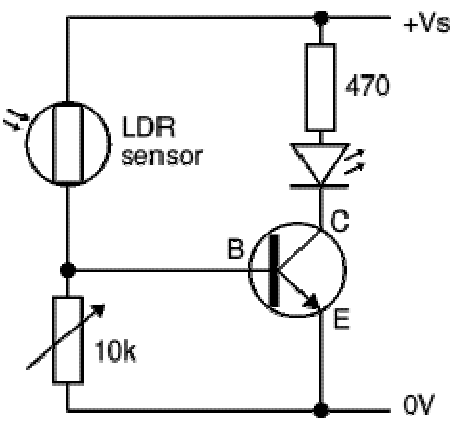
```
{
  unsigned char i, j;
  unsigned int A2D,A2D0;
  unsigned int HUMIDITY;

  HUMIDITY = 0;
  VOLT = 0;

while(1) {

    A2D0 = A2D_Read(0);
    A2D1 = A2D_Read(1);

    HUMIDITY = A2D1*0.097;


   //Display Humidity
    LCD_Move(2,10);
    HUMIDITY = (HUMIDITY)*100;
    LCD_Out2(HUMIDITY);


    SCI_Out(HUMIDITY, 2);
    SCI_CRLF();


    Wait_ms(1000);
```

Light sensor



LDRs or Light Dependent Resistors are very useful especially in light/dark sensor circuits. Normally the resistance of an LDR is very high, sometimes as high as 1000 000 ohms, but when they are illuminated with light resistance drops dramatically.

Schematic:



Project comments:

PROBLEMS FACED:

- Even though the PCB had originally been designed to have its lower side as a ground plane, it still need a wire that touches the lower plane and goes to a GND plug-in (we now use the ground plug-in provided in the PIC).
- Transferring the common schematic from Multisim to Ultiborad was a huge problem, we had to start from scratch in Ultiboard in order to ensure that we have all our parts realized in real practice.
- We had to use de-coupling capacitors with the power supply.
- Realizing that the common power supply that we had to use was 5V, we had to change some of the sensors including humidity sensor, wind sensor whose original circuit needed a 12V supply.
- Creating custom footprints for all the parts was a long and cumbersome process.
- Making the final program that used all the sensors was very challenging, especially to come up with a successful build in MPLAB.
- We had to get 3 PCB's made for this, the third one being the successful one
- Even after we had a successful PCB, after soldering in all the parts we had to check for short circuit and wrong traces on the PCB.
- The wind sensor was never shipped to us by the company, which is one of the reasons we don't have a wind sensor on board, even though we have a slot for it. One we attain it, all that needs to be done is soldering and a slight modification in the program.

FUTURE IMPROVEMENTS:

- After the wind sensor is obtained, it should be soldered immediately in the slot made for it in the PCB, make sure you take a good look at the specifications of the wind sensor. The program for it should be fairly easy to compile as it is a fairly smart sensor.

- The encasing needs to be changed if this model is to be used during winter and really harsh conditions.
- Modifying the PIC, custom designing it especially for this so as to accommodate all the different values being measured on the LCD display screen.

TROUBLE SHOOTING:

TEMPERATURE SENSOR:

Temperature sensor should get us 2.98V in the output pin of the AMP04 amplifier in 25C if its not giving us the appropriate temperature we should measure the voltages at pin 3 and 2 of the AMP04.

The voltage at 3 should be 2.9 and voltage at pin 2 should be less than pin 3 coz if we need a high output we need to have voltage higher than the voltage at pin 2 and with the help of multimeter we should be able to check the voltage at output of AMP04 pin 6 and with the help of 10 k pot we can also adjust the voltage at pin 6.Best way to go about it is we should check the room temperature suppose 23C multiply it by .01we get .23 and add it to 2.73 comes out to be 2.96 and with the help of the pot try to adjust the temperature.

HUMIDITY SENSOR:

This sensor is pretty expensive and very nice. Easy to calibrate don't have to worry about the pot. We should not connect the power to pot that how it works. and the output is kind of linear. The best way to go about it in room condition it gives us .4 which is suppose to give us 2 volt in the output of 6^{th} pin of AMP04. If it is not that we should sense something Is goofy and try to check the voltage at pin and try to multiply it by 5 which is the voltage at pin 6 of AMP04.

LIGHT SENSOR:

If something goofy occurs in light sensor must be something wrong with the transistor or may be light sensor is smoked we need to fix it only by replacing it.

If no data is being recorded by the PIC:

- One of the reasons for this could be that the ground in the PCB is not connected to the ground plug-in in the PIC, make sure that the wire that touches the bottom half of the PCB (the side that only has the day/night sensor soldered on it) is connected to the plug-in on the PIC that says GND. Make sure that color coding of all the wires is correct.
- If the above mentioned step still does not produce any result, that means that something might be wrong with the PCB, take the PCB out of the

encasing and check for short-circuit using a multi-meter alongside each of the sensors, making sure each of them is giving some kind of output. If for some reason one of them fails to give an output, then this would be an indication that this particular part has to be replaced.

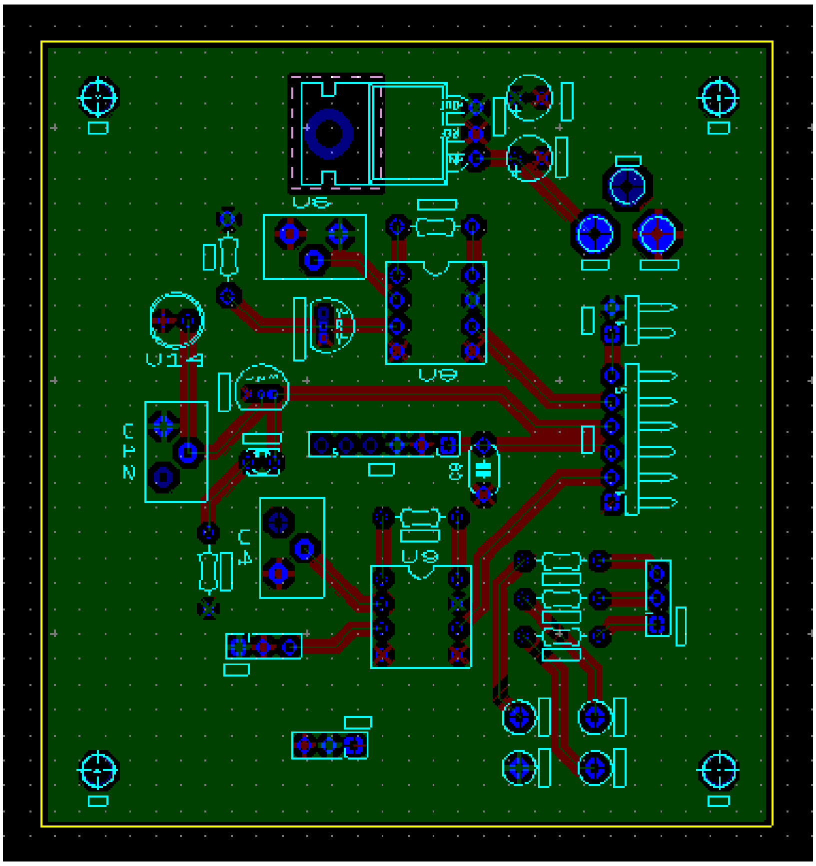
If the PIC displays a blank screen:

- One of the chief reasons for this would be that somehow the PIC has been reset, and this would mean that you would have to reload the program in the PIC once again. This can be done using the steps as illustrated in previous sections.

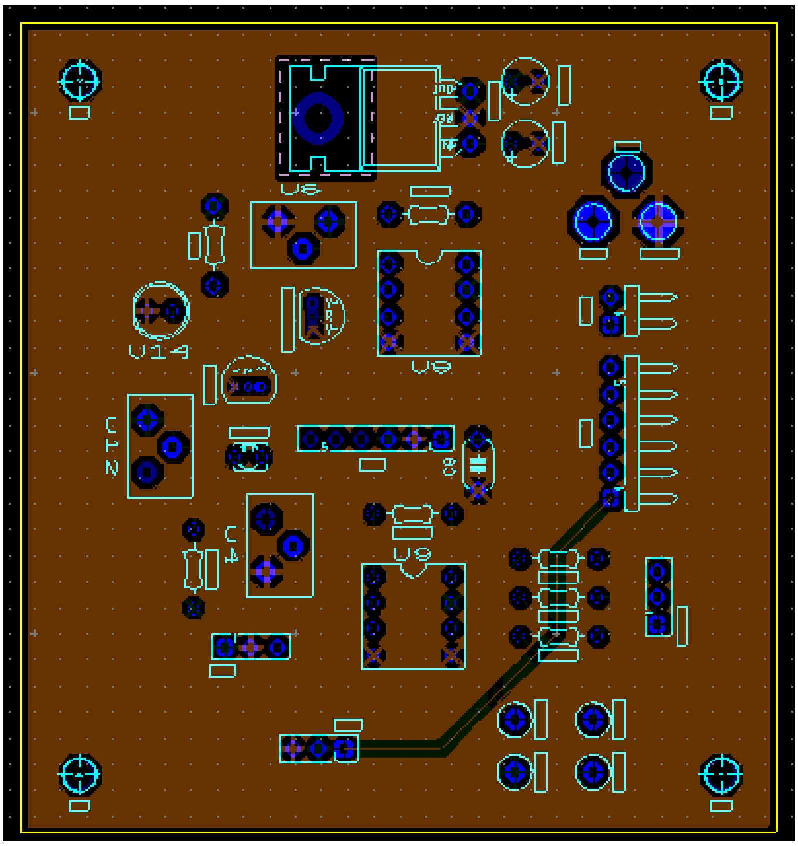
APPENDIX

PCB layout:

Upper side:



Lower side:



Software Program:

```
// Global Variables

const unsigned char MSG0[21] = "Pressure:           ";
const unsigned char MSG1[21] = "Celcius:            ";
const unsigned char MSG2[21] = "Humidity:           ";
const unsigned char MSG3[21] = "Rain:               ";


 unsigned int RAIN;

// Subroutine Declarations
#include   <pic18.h>
#include        "lcd_portd.h"
#include        "function.h"

// Subroutines
#include        "lcd_portd.c"
#include        "function.c"


void A2D_Init(void)
{
   TRISA = 0xFF;
   TRISE = 0x0F;
   ADCON2 = 0x95;
   ADCON1 = 0x07;
   ADCON0 = 0x01;
    }

unsigned int A2D_Read(unsigned char c)
{
unsigned int result;
unsigned char i;
c = c & 0x0F;
ADCON0 = (c << 2) + 0x01; // set Channel Select
for (i=0; i<3; i++); // wait 2.4us (approx)
GODONE = 1; // start the A/D conversion
while(GODONE); // wait until complete (approx 8us)
return(ADRES);
}


   void SCI_Init(void)
   {
    TRISC = TRISC | 0xC0;
    TXIE = 0;
    RCIE = 0;
```

```
    BRGH = 1;
    SPBRG = 31;
    TXSTA = 0x22;
    RCSTA = 0x90;
  }

void SCI_Out(unsigned int DATA, unsigned char N)
{
   unsigned char A[5], i;
   for (i=0; i<5; i++) {
      A[i] = DATA % 10;
      DATA = DATA / 10;
      }
   for (i=5; i>0; i--) {
      if (i == N) {
         while(!TRMT);  TXREG = '.';
      }
      while(!TRMT);  TXREG = ascii(A[i-1]);
   }

   while(!TRMT);  TXREG = ' ';
   }

  void SCI_CRLF(void)
  {
   while(!TRMT);  TXREG = 13;
   while(!TRMT);  TXREG = 10;
  }


   void Display(unsigned int T)
   {
   unsigned char A[5], i;
   if (T < 0) {
   T = -T;
   LCD_Write('-');
   }
   else LCD_Write('+');
   for (i=0; i<5; i++) {
   A[i] = T % 10;
   T = T / 10;
    }
   for (i=5; i>0; i--) {
   LCD_Write(A[i-1] + '0');
   if (i == 3) LCD_Write('.');
   }
}


  void LCD_Out(int T)
  {
  unsigned char A[5], i;
```

```
  if (T < 0) {
  T = -T;
  LCD_Write('-');
  }
  else LCD_Write('+');
  for (i=0; i<5; i++) {
  A[i] = T % 10;
  T = T / 10;
  }
  for (i=5; i>0; i--) {
  LCD_Write(A[i-1] + '0');
  if (i == 3) LCD_Write('.');
  }
  }


   void LCD_Out2(int T)
{
   unsigned char A[5], i;
   if (T < 0) {
   T = -T;
   LCD_Write('-');
}
   else LCD_Write('+');
   for (i=0; i<5; i++) {
   A[i] = T % 10;
   T = T / 10;
}
   for (i=5; i>0; i--) {
   LCD_Write(A[i-1] + '0');
   if (i == 3) LCD_Write('.');
}
}


   void LCD_Out3(int T)
{
   unsigned char A[5], i;
   if (T < 0) {
   T = -T;
   LCD_Write('-');
}
   else LCD_Write('+');
   for (i=0; i<5; i++) {
   A[i] = T % 10;
  T = T / 10;
}
  for (i=5; i>0; i--) {
  LCD_Write(A[i-1] + '0');
  if (i == 3) LCD_Write('.');
}
}
```

```
         void interrupt IntServe(void) {
          if (INT0IF) {
            if (RB0) RAIN += 1;
            INT0IF = 0;
           }
        }


// Main Routine

void main(void)
{
   unsigned char i, j;
   unsigned int A2D,A2D0,A2D1,A2D2, RED, GREEN, BLUE,VOLT;
   unsigned int CELCIUS;
   unsigned int HUMIDITY;
   unsigned int PRESSURE;

   PRESSURE = 0;
   CELCIUS = 0;
   HUMIDITY = 0;
   VOLT = 0;

   TRISC = 0;
   TRISB = 0;
   TRISB0 = 1;
   ADCON1 = 15;
   INTEDG0 = 1; // rising edges

   RAIN = 0;

   LCD_Init();                  // initialize the LCD
   A2D_Init();
   SCI_Init();

   LCD_Move(0,0);   for (i=0; i<20; i++) LCD_Write(MSG0[i]);
   LCD_Move(1,0);   for (i=0; i<20; i++) LCD_Write(MSG1[i]);
   LCD_Move(2,0);   for (i=0; i<20; i++) LCD_Write(MSG2[i]);
   LCD_Move(3,0);   for (i=0; i<20; i++) LCD_Write(MSG3[i]);

   INT0IE = 1;
   GIE = 1;
   PEIE = 1;

   while(1) {

      A2D0 = A2D_Read(0);
      A2D1 = A2D_Read(1);
      A2D2 = A2D_Read(2);
```

```
      VOLT = A2D2*.488;
      CELCIUS=A2D0*0.048;
      HUMIDITY = A2D1*0.097;
      PRESSURE = 22.22*VOLT + 10.56;

      //Display Pressure
      LCD_Move(0,10);
      LCD_Out(PRESSURE);

      //Display Celcius
      LCD_Move(1,10);
      CELCIUS = (CELCIUS)*100;
      Display(CELCIUS);

      //Display Humidity
       LCD_Move(2,10);
       HUMIDITY = (HUMIDITY)*100;
       LCD_Out2(HUMIDITY);

       LCD_Move(3,10);
       LCD_Out(RAIN);

       SCI_Out(PRESSURE, 2);
       SCI_Out(CELCIUS, 2);
       SCI_Out(HUMIDITY, 2);
       SCI_Out(RAIN, 2);
       SCI_CRLF();


       Wait_ms(1000);


if (CELCIUS < 15)
{
      BLUE = 100;
      RED = 0;
      GREEN = 0;
      if ((CELCIUS >= 15) & (CELCIUS < 25) )
{
      GREEN = 10*(CELCIUS - 15);
      BLUE = 100 - GREEN;
      RED = 0;
  }
      if ((CELCIUS >= 25) & (CELCIUS < 40) )
{
      RED = 10*(CELCIUS - 25);
      GREEN = 100 - RED;
      BLUE = 0;
  }
      if (CELCIUS >= 40)
```

```
 {
           RED = 100;
      GREEN = 0;
      BLUE = 0;
  }
      for (j=0; j<100; j++)
{
      for (i=0; i<100; i++)
 {
      if (RED > i) RC0 = 0; else RB1 = 1;
      if (GREEN > i) RC1 = 0; else RB2 = 1;
      if (BLUE > i) RC2 = 0; else RB3 = 1;
  }
  }
    }
   }
}
```

Pressure sensor:

Model No.- MPX4115A-ND   Price per unit-10.34$

Media Resistant, Integrated Silicon Pressure Sensor for Manifold Absolute Pressure, Altimeter or Barometer Applications On-Chip Signal Conditioned, Temperature Compensated and Calibrated

The MPX4115A series is designed to sense absolute air pressure in altimeter or barometer (BAP) applications. Freescale's BAP sensor integrates on-chip, bipolar op amp circuitry and thin film resistor networks to provide a high level analog output signal and temperature compensation. The small form factor and high reliability of on-chip integration makes the Freescale BAP sensor a logical and economical choice for application designers.

Features

- 1.5% Maximum Error Over 0° to 85°C
- Ideally Suited for Microprocessor Interfacing or Microcontroller-Based Systems
- Temperature Compensated Over -40°C to +125°C
- Durable Epoxy Unibody Element or Thermoplastic (PPS) Surface Mount Package
- Available as Standard Fluorosilicone Gel (MPXA4115A, MPX4115A) or Media Resistant Gel (MPXAZ4115A)

Go to: http://www.freescale.com/files/sensors/doc/data_sheet/MPX4115A.pdf

Temperature Sensor:

Model no- LM-335-ND   price per unit- 1.43$

Humidity sensor:

Model no- CHS-GSS   price per unit- 37 $

CHS series humidity sensors are compact and extremely simple to apply. Because they contain the necessary circuitry, there is no need to provide additional control circuitry or perform time-consuming calibration. With simple connection to a power supply, they will output DC at 100% relative humidity. This makes it possible to read RH directly with a voltmeter.

CHS-U TYPE

For industrial use and measuring equipment

FEATURES

- These sensors can measure a wide range of humidity – from 5(%) to 95(%)RH.
- They are highly accurate. The nominal accuracy for the CHSUPR and CHR-UPS is within ±3(%) RH.
- Characteristics are stable over a wide temperature range.
- Humidity sensing characteristics exhibit virtually no hysteresis.
- Highly cost-effective and compact, requiring extremely little mounting space.

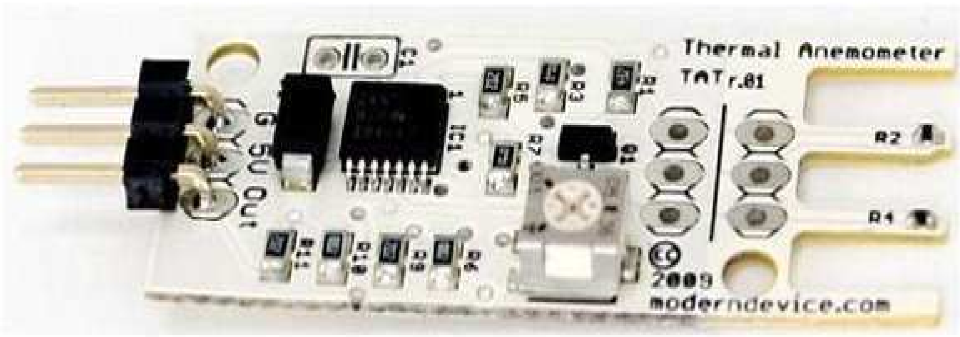
• Low current consumption.
• Outputs DC.1V at 100(%)RH; relative humidity can be read directly with a voltmeter.
• All-in-one construction integrates sensor with support circuitry. The entire module operates off a 5V power supply.
• Generated ripple at low humidity levels will not exceed 2.5mV.

Go to: http://datasheet.octopart.com/CHS-GSS-TDK-datasheet-130906.pdf

Wind sensor:

Price per unit-17 $

Model- It is only manufactured by that particular company



Go to: http://www.moderndevice.com/products/wind-sensor

Rain-gauge:

Price per unit – 72.95 $

Go to: http://www.rainwise.com/products/attachments/6697/20070415082348.pdf

PIC controller used:

Model no: PIC18F4620 price per unit: 85 $

Amplifier used:

Model no- AMP04 Price per unit: 10$

Quantity-2

Description:

FEATURES
Single Supply Operation
Low Supply Current: 700 _A Max
Wide Gain Range: 1 to 1000
Low Offset Voltage: 150 _V Max
Zero-In/Zero-Out
Single-Resistor Gain Set
8-Lead Mini-DIP and SO Packages
APPLICATIONS
Strain Gages
Thermocouples
RTDs
Battery-Powered Equipment
Medical Instrumentation

Data Acquisition Systems
PC-Based Instruments
Portable Instrumentation
Precision Single Supply
Instrumentation Amplifier
GENERAL DESCRIPTION
The AMP04 is a single-supply instrumentation amplifier designed to work over a +5 volt to }15 volt supply range. It offers an excellent combination of accuracy, low power consumption, wide input voltage range, and excellent gain performance.
Gain is set by a single external resistor and can be from 1 to 1000. Input common-mode voltage range allows the AMP04 to handle signals with full accuracy from ground to within 1 volt of the positive supply. And the output can swing to within 1 volt of the positive supply. Gain bandwidth is over 700 kHz. In addition to being easy to use, the AMP04 draws only 700 µA of supply current.
For high resolution data acquisition systems, laser trimming of low drift thin-film resistors limits the input offset voltage to under 150 µV, and allows the AMP04 to offer gain nonlinearity of 0.005% and a gain tempco of 30 ppm/°C.
A proprietary input structure limits input offset currents to less than 5 nA with drift of only 8 pA/°C, allowing direct connection of the AMP04 to high impedance transducers and other signal sources.
The AMP04 is specified over the extended industrial (–40°C to +85°C) temperature range. AMP04s are available in plastic and ceramic DIP plus SO-8 surface mount packages.
Contact your local sales office for MIL-STD-883 data sheet and availability.

Go to: http://www.analog.com/static/imported-files/data_sheets/amp04.pdf

Power cable:

Model no-155601 Price-9.95 $

Go to: http://www.jameco.com/Jameco/Products/ProdDS/155601.pdf

Other parts Used:

- Connecting wires
- Acrylic glass
- 8 nuts,4 bolts,8 washers
- 4 PVC 9” long pipes
- Velcro

- plastic stand-offs with 4 nuts that can be accommodated

Total Price: 253.67 $ (Excluding the price required to make the encasing and standard RLC components, potentiometer)